# 【第02-01章】【NIO入门】

## 传统的BIO 编程

网络编程的基本模型是Client/Server 模型，也就是两个进程之间进行相互通信，其中服务端提供位置信息（绑定的IP 地址和监听端口），客户端通过连接操作向服务端监听的地址发起连接请求，通过三次握手建立连接，如果连接建在成功，双方就可以通过网络套接字（ Socket ）进行通信。

在基于传统同步阻塞模型开发中， ServerSocket 负责绑定IP 地址，启动监听端口；Socket负责发起连接操作。连接成功之后，双方通过输入和输出流进行同步阻塞式通信。

## BIO 通信模型图

BIO的服务端通信棋型： 采用BIO 通信模型的服务端，通常由一个独立的Acceptor 线程负责监听客户端的连接，它接收到客户端连接请求之后为每个客户端创建一个新的线程进行链路处理，处理完成之后，通过输出流返回应答给客户端，线程销毁。这就是典型的一请求一出答通信模型。



图1 BIO 通信模型图

该模型最大的问题就是缺乏弹性伸缩能力，当客户端并发访问量增加后，服务端的线程个数和客户端井发访问数里l : I 的正比关系，由于线程是Java 虚拟机非常宝贵的系统资源， 当线程数膨胀之后，系统的性能将急剧下降，随着并发访问握的继续增大，系统会发生线程堆栈溢出、创建新线程失败等问题，并最终导致进程宕机或者僵死， 不能对外提供服务。